

# On shortest disjoint paths in planar graphs<sup>☆</sup>

Yusuke Kobayashi<sup>a,\*</sup>, Christian Sommer<sup>b</sup>

<sup>a</sup> Department of Mathematical Informatics, Graduate School of Information Science and Technology, University of Tokyo, Tokyo 113-8656, Japan

<sup>b</sup> Department of Computer Science, Graduate School of Information Science and Technology, University of Tokyo, Tokyo 113-0033, Japan

## ARTICLE INFO

### Article history:

Received 12 November 2009

Received in revised form 8 April 2010

Accepted 21 May 2010

Available online 15 June 2010

### MSC:

05C85

68R10

90C27

90C35

### Keywords:

Disjoint paths

Shortest path

Planar graph

Objective function

Minimum cost flow

## ABSTRACT

For a graph  $G$  and a collection of vertex pairs  $\{(s_1, t_1), \dots, (s_k, t_k)\}$ , the  $k$  disjoint paths problem is to find  $k$  vertex-disjoint paths  $P_1, \dots, P_k$ , where  $P_i$  is a path from  $s_i$  to  $t_i$  for each  $i = 1, \dots, k$ . In the corresponding optimization problem, the shortest disjoint paths problem, the vertex-disjoint paths  $P_i$  have to be chosen such that a given objective function is minimized. We consider two different objectives, namely minimizing the total path length (minimum sum, or short: Min-Sum), and minimizing the length of the longest path (Min-Max), for  $k = 2, 3$ .

**Min-Sum:** We extend recent results by Colin de Verdière and Schrijver to prove that, for a planar graph and for terminals adjacent to at most two faces, the Min-Sum 2 Disjoint Paths Problem can be solved in polynomial time. We also prove that, for six terminals adjacent to one face in any order, the Min-Sum 3 Disjoint Paths Problem can be solved in polynomial time.

**Min-Max:** The Min-Max 2 Disjoint Paths Problem is known to be **NP**-hard for general graphs. We present an algorithm that solves the problem for graphs with tree-width 2 in polynomial time. We thus close the gap between easy and hard instances, since the problem is weakly **NP**-hard for graphs with tree-width 3.

© 2010 Elsevier B.V. All rights reserved.

## 1. Introduction

The *vertex-disjoint paths problem* is one of the classic problems in algorithmic graph theory and combinatorial optimization, and has many applications, for example in transportation networks, VLSI-design [1,2], or routing in networks [3,4]. The input of the vertex-disjoint paths problem is a graph  $G = (V, E)$  and  $k$  pairs of vertices  $(s_1, t_1), \dots, (s_k, t_k)$ , for which the algorithm has to find  $k$  pairwise vertex-disjoint paths connecting  $s_i$  and  $t_i$ , if they exist. Paths are called *vertex-disjoint* if they have no vertices in common.

In the optimization version of the problem, we are interested in *short* vertex-disjoint paths. We may want to minimize the total length (minimum sum) or the length of the longest path (Min-Max objective function). A more formal description of the problem is as follows.

### Min-Sum $k$ Disjoint Paths Problem (Min-Max $k$ Disjoint Paths Problem)

**Input:** A graph  $G = (V, E)$ ,  $k$  pairs of vertices  $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$  in  $G$  (which are sometimes called *terminals*), and a length function  $l : E \rightarrow \mathbb{R}_+$ .

**Output:** Vertex-disjoint paths  $P_1, \dots, P_k$  in  $G$  such that  $P_i$  is from  $s_i$  to  $t_i$  for  $i = 1, 2, \dots, k$ , minimizing  $\sum_{i=1}^k l(P_i)$  (or minimizing  $\max_i l(P_i)$ ), where  $l(P_i) = \sum_{e \in E(P_i)} l(e)$ .

<sup>☆</sup> A preliminary version of this work appears in *Proceedings of International Symposium on Algorithms and Computation, ISAAC 2009*, December 16–18, 2009, pp. 293–302.

\* Corresponding author.

E-mail addresses: [kobayashi@mist.i.u-tokyo.ac.jp](mailto:kobayashi@mist.i.u-tokyo.ac.jp) (Y. Kobayashi), [sommer@is.s.u-tokyo.ac.jp](mailto:sommer@is.s.u-tokyo.ac.jp) (C. Sommer).

**Table 1**  
Complexity of the vertex-disjoint paths problem.

	Undirected version	Directed version
$k$ : constant	<b>P</b> [12]	<b>NP-hard</b> [8] (Planar digraph: <b>P</b> [7]) (Acyclic digraph: <b>P</b> [8])
$k$ : general	<b>NP-hard</b> [5] (Planar graph: <b>NP-hard</b> [6])	<b>NP-hard</b> [5] (Planar digraph: <b>NP-hard</b> [6])

**Table 2**  
Results for the Min-Sum Disjoint Paths Problem.

	Conditions	Complexity
$k = 2$	Directed Directed, planar, one face Undirected Undirected, planar, two faces	<b>NP-hard</b> [8] OPEN OPEN <b>P</b> (Theorem 2)
$k = 3$	Undirected, planar, one face	<b>P</b> (Theorem 11)
$k$ : fixed	Undirected	OPEN
$k$ : general	Undirected $s_1 = \dots = s_k$ and/or $t_1 = \dots = t_k$ Planar, one face, well-ordered Planar, $S \neq T$ faces Bounded tree-width	<b>NP-hard</b> [5] <b>P</b> (Min-cost flow) <b>P</b> (Min-cost flow) <b>P</b> [13] $O(n)$ [14]

### 1.1. Related work

If  $k$  is part of the input, the vertex-disjoint paths problem is one of Karp's **NP-hard** problems [5], and it remains **NP-hard** even if  $G$  is constrained to be planar [6]. If  $k$  is a fixed number,  $k$  pairwise vertex-disjoint paths can be found in polynomial time in directed planar graphs [7] and in directed acyclic graphs [8], whereas the problem in general directed graphs is **NP-hard** even if  $k = 2$  [8]. It is known that the disjoint paths problem in undirected graphs is solvable in polynomial time when  $k = 2$  [9–11]. Perhaps the biggest achievement in this area is Robertson and Seymour's polynomial-time algorithm for the problem in undirected graphs when  $k$  is fixed [12]. We summarize known results on the problems in Table 1.

The optimization problem is considerably harder. The problem of finding disjoint paths minimizing the total length is wide open and only a few cases are known to be solvable in polynomial time (see also Table 2). First, finding  $k$  internally disjoint  $s$ – $t$  paths (i.e.,  $s_1 = \dots = s_k = s$  and  $t_1 = \dots = t_k = t$ ) with minimum total length (*Min-Sum*) is still possible in polynomial time, since it reduces to finding the standard minimum cost flow. The Min-Sum problem is solvable in linear time for graphs with bounded tree-width [14]. For the following two cases, the Min-Sum problem can also be reduced to the minimum cost flow problem and can thus be solved in polynomial time:

- All sources (or sinks) coincide, that is,  $s_1 = \dots = s_k$  (or  $t_1 = \dots = t_k$ , respectively).
- The graph is planar, all terminals are incident with a common face, and their cyclic order is  $s_1, \dots, s_k, t_k, \dots, t_1$  (called *well-ordered*).

Another special case of the Min-Sum problem has recently been solved by Colin de Verdière and Schrijver [13]. They showed the following:

**Theorem 1** ([13]). *If a given directed or undirected graph  $G$  is planar, all sources are incident to one face  $S$ , and all sinks are incident to another face  $T \neq S$ , then we can find  $k$  vertex-disjoint paths in  $G$  with minimum total length in  $O(kn \log n)$  time.*

If the length of the longest path is to be minimized (*Min-Max*), the problem seems to be harder than the Min-Sum problem. The problem of finding two internally disjoint  $s$ – $t$  paths minimizing the length of the longer path is **NP-hard** for an acyclic directed graph [15], but 2-approximable [16] using the *Min-Sum* version. Moreover, the problem is strongly **NP-hard** for general directed graphs when  $s_1, s_2, t_1$ , and  $t_2$  are distinct [16]. For an overview, see Table 3.

Yet another variant of the objective function for the problem of finding two disjoint paths for one pair of terminals is the following: Before summing up the path lengths, the length of the longer path is multiplied by a factor  $\alpha \in (0, 1)$ , which parameterizes the cost function.  $\alpha = 1$  would yield the Min-Sum variant and  $\alpha = 0$  would yield the so-called ‘min-min’ variant, in which the length of the shorter path is to be minimized, which is **NP-hard** [18]. For  $\alpha \in (0, 1)$  there is an approximation algorithm with ratio  $\frac{1+\alpha}{2\alpha}$ , which, for directed graphs, is claimed to be optimal unless the polynomial hierarchy collapses completely [19]. If the length of the shorter path is multiplied by  $\alpha \in (0, 1)$ , there is an approximation algorithm with ratio  $\frac{2}{1+\alpha}$ , which is claimed to be tight as well [17].

**Table 3**

Results for the Min-Max Disjoint Paths Problem.

	Conditions	Complexity
$k = 2$	Directed, acyclic, $s_1 = s_2, t_1 = t_2$	<b>NP</b> -hard [15]
	Directed, $s_1 = s_2, t_1 = t_2$	Pseudo-polynomial [16]
	Directed	2-approx. [16]
	Undirected, tree-width = 3, planar	Strongly <b>NP</b> -hard [16]
	Undirected, tree-width $\leq 2$	<b>NP</b> -hard ([17] and Theorem 14) <b>P</b> (Theorem 18)

**Table 4**

Min-sum results in planar undirected graphs.

	One face	Two faces
$k = 2$	Flow	Theorem 1, Propositions 3 and 6
$k = 3$	Flow, Theorem 11	OPEN

## 1.2. Contribution

We extend the Min-Sum results of Colin de Verdière and Schrijver [13] for undirected graphs and  $k = 2$  as follows: the two disjoint faces  $F_1, F_2$  may be ‘mixed’ such that

- $s_1, s_2, t_1$  are incident to  $F_1$  and  $t_2$  is incident to  $F_2$  (Proposition 3),
- $s_1, t_1$  are incident to  $F_1$  and  $s_2, t_2$  are incident to  $F_2$  (Proposition 6).

Our algorithms consist of non-trivial reductions to Theorem 1. By combining Theorem 1 with our new results, flow reductions, and trivially infeasible inputs, we obtain the following theorem.

**Theorem 2.** *Let  $G = (V, E)$  be an undirected planar graph, and let  $F_1$  and  $F_2$  be two of its faces. If each terminal is on one of the boundaries of  $F_1$  and  $F_2$ , then the Min-Sum 2 Disjoint Paths Problem in  $G$  is solvable in polynomial time.*

We also give a polynomial-time algorithm for the Min-Sum Disjoint Paths Problem when  $k = 3$  and all terminals are incident to one face (Theorem 11). Our contribution is to give an algorithm for the case when the terminals are not well-ordered, by a non-trivial reduction to Theorem 1. For a summary, see Table 4.

For the Min-Max 2 Disjoint Paths Problem, we draw the line between tractable and hard problems: We prove weak **NP**-hardness of the Min-Max 2 Disjoint Paths Problem for planar graphs with tree-width 3 using a reduction from the PARTITION problem (Theorem 14). We later learned that the reduction was used independently in almost the same manner in [17] already, without an explicit link to the tree-width and the Min-Max variant. For graphs with tree-width 2 (including series-parallel graphs and outer-planar graphs), we provide a polynomial-time algorithm (Theorem 18). The same algorithm also works for the Min-Min 2 Disjoint Paths Problem and the  $\alpha$ -variants from [17, 19].

For both the Min-Sum and the Min-Max versions and for the variants with cost functions parameterized by  $\alpha$  as defined in [17, 19], we give a pseudo-polynomial-time algorithm for graphs with bounded tree-width (Theorem 19). The algorithm runs in polynomial time for the Min-Sum objective function [14].

## 2. Preliminaries

Let  $G = (V, E)$  be an undirected graph with vertex set  $V$  and edge set  $E$ , and let  $n = |V|$  denote the number of vertices. Since we consider vertex-disjoint paths, in what follows, we may assume that the graph has no multiple edges and no self-loops. An edge connecting  $u, v \in V$  is denoted by  $uv$ , whereas  $(u, v)$  represents the arc from  $u$  to  $v$  in a directed graph. For a subgraph  $H$  of  $G$ , the vertex set and the edge set of  $H$  are denoted by  $V(H)$  and  $E(H)$ , respectively. Let  $\delta(v)$  denote the set of edges incident to  $v \in V$ . For  $U \subseteq V$ , let  $G[U]$  be the subgraph of  $G = (V, E)$  induced by  $U$ , that is, its vertex set is  $U$  and its edge set consists of all edges in  $E$  with both ends in  $U$ . A graph  $G$  is *planar* if it can be embedded in a plane  $\Sigma$  such that no edges intersect, except at their end points. To simplify notation, we do not distinguish between a vertex of  $G$  and the point of  $\Sigma$  used in the embedding to represent the vertex, and we do not distinguish between an edge and the curve on  $\Sigma$  representing it. A *region* is a subset of  $\Sigma$ , and for a region  $R$ , let  $G[R]$  denote the subgraph of  $G$  consisting of the vertices and the edges in  $R$ . For a face  $F$  of a planar graph, let  $\partial F$  denote the boundary of  $F$ . A planar graph is *outer-planar* if it allows for a planar embedding such that all its vertices are on the outer face. A *path*  $P$ , which is denoted by  $P = (v_1, v_2, \dots, v_l)$ , is a sequence consisting of vertices  $v_1, \dots, v_l$  and edges  $e_1 = v_1v_2, \dots, e_{l-1} = v_{l-1}v_l$ . When  $v_1 = v_l$ , it is called a *cycle*. A path (or a cycle) is *simple* if  $v_i \neq v_j$  for distinct  $i, j$  (except for  $v_1 = v_l$ ). For a simple path  $P = (v_1, v_2, \dots, v_l)$ , let  $P^{[v_i, v_j]}$  denote the path  $(v_i, v_{i+1}, \dots, v_j)$ , and it is called a *subpath* of  $P$ . For a length function  $l : E \rightarrow \mathbb{R}_+$ , the length of a path  $P$  is denoted by  $l(P)$ , and for a pair of vertices  $u, v \in V$ , let  $d_G(u, v)$  denote the length of a shortest path connecting  $u$  and  $v$  in  $G$ .

Suppose that two simple paths  $P_1 = (v_1^1, v_2^1, \dots, v_{l_1}^1)$  and  $P_2 = (v_1^2, v_2^2, \dots, v_{l_2}^2)$  in a planar graph have a common vertex  $v_i^1 = v_j^2 = v$ . We say that  $P_1$  and  $P_2$  *cross at  $v$*  if  $vv_{i-1}^1, vv_{j-1}^2, vv_{i+1}^1$ , and  $v, v_{j+1}^2$  are incident with  $v$  cyclically in this order.

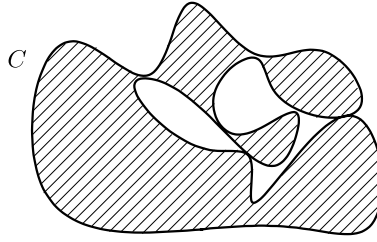


Fig. 1. Inside of a cycle.

Similarly, suppose that two paths  $P_1$  and  $P_2$  have a common subpath  $P$  in a planar graph. Assume that if we contract all edges in  $P$ , then the two paths corresponding to  $P_1$  and  $P_2$  cross at the vertex corresponding to  $P$ . In this case, we say that  $P_1$  and  $P_2$  cross at  $P$ . We say that a path or a cycle crosses with itself if it has two crossing subpaths.

For a simple cycle  $C$  in a planar graph, the *inside* of  $C$  is the bounded closed region whose boundary is  $C$ . We define the inside of  $C$  when  $C$  is not simple but does not cross with itself (see Fig. 1).

The tree-width of a graph was introduced by Halin [20], but it went unnoticed until it was rediscovered by Robertson and Seymour [21] and, independently, by Arnborg and Proskurowski [22]. The *tree-width* of a graph is defined as follows.

**Definition 1.** Let  $G$  be a graph,  $T$  a tree and let  $\mathcal{V} = \{V_t \subseteq V(G) \mid t \in V(T)\}$  be a family of vertex sets of  $G$  indexed by the vertices  $t$  of  $T$ . The pair  $(T, \mathcal{V})$  is called a *tree-decomposition* of  $G$  if it satisfies the following three conditions:

- $V(G) = \bigcup_{t \in T} V_t$
- for every edge  $e \in G$  there exists a  $t \in T$  such that both ends of  $e$  lie in  $V_t$
- If  $t, t', t'' \in V(T)$  and  $t'$  lies on the path of  $T$  between  $t$  and  $t''$ , then  $V_t \cap V_{t''} \subseteq V_{t'}$ .

The *width* of  $(T, \mathcal{V})$  is the number  $\max\{|V_t| - 1 \mid t \in T\}$  and the *tree-width*  $\text{tw}(G)$  of  $G$  is the minimum width of any tree-decomposition of  $G$ .

The tree-width is a good measure of the algorithmic tractability of graphs. It is known that a number of hard problems on graphs, such as “Hamiltonian cycle” and “chromatic number”, can be solved efficiently when the given graph has small tree-width [22]. A graph has tree-width 1 if and only if it is a forest, and families of graphs with tree-width at most 2 include outer-planar graphs and series-parallel graphs.

### 3. Min-Sum objective function

In this section, we deal with the Min-Sum  $k$  Disjoint Paths Problem for  $k = 2, 3$ . To simplify the arguments, we assume that each non-self-crossing path has a different length, and in particular, the Min-Sum  $k$  Disjoint Paths Problem has a unique optimal solution if it has a feasible solution. Actually, we do not need this assumption if we choose an appropriate shortest path in each step of the proof, but we omit the detailed description.

Note that we can also use a perturbation technique such that all shortest paths are unique (see [23]). For  $E = \{e_1, e_2, \dots, e_m\}$  and  $l : E \rightarrow \mathbb{R}_+$ , we use a new length function  $l' : E \rightarrow \mathbb{R}_+$  defined by  $l'(e_i) = l(e_i) + \varepsilon^i$  for each  $i$ , where  $\varepsilon$  is an infinitely small positive number. Then, each path has a different length, but this technique increases the running time, since the comparison of two lengths requires  $O(m)$  time.

#### 3.1. Min-Sum 2 Disjoint Paths Problem

In this section, we prove Theorem 2, which we restate here.

**Theorem.** Let  $G = (V, E)$  be an undirected planar graph, and let  $F_1$  and  $F_2$  be two of its faces. If every terminal is on  $\partial F_1 \cup \partial F_2$ , then the Min-Sum 2 Disjoint Paths Problem in  $G$  is solvable in polynomial time.

**Proof.** The four terminals  $s_1, s_2, t_1$ , and  $t_2$  may lie on two faces as follows:

- $s_1, s_2, t_1$ , and  $t_2$  are incident to  $F_1$  (min-cost flow or trivially infeasible),
- $s_1, s_2$  are incident to  $F_1$  and  $t_1, t_2$  are incident to  $F_2$  (Theorem 1 due to [13]),
- $s_1, s_2, t_1$  are incident to  $F_1$  and  $t_2$  is incident to  $F_2$  (Proposition 3), or
- $s_1, t_1$  are incident to  $F_1$  and  $s_2, t_2$  are incident to  $F_2$  (Proposition 6).

The remaining cases (e.g. the case with  $s_2$  alone on one face) are symmetric for undirected graphs.  $\square$

**Proposition 3.** Let  $G = (V, E)$  be an undirected planar graph, and  $F_1$  and  $F_2$  be two of its faces. If three terminals are on  $\partial F_1$  and the remaining terminal is on  $\partial F_2$ , then the Min-Sum 2 Disjoint Paths Problem in  $G$  is solvable in  $O(n^3 \log n)$  time.

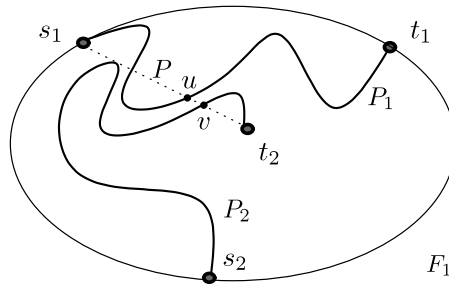


Fig. 2. Definitions of  $P$ ,  $u$ , and  $v$ .

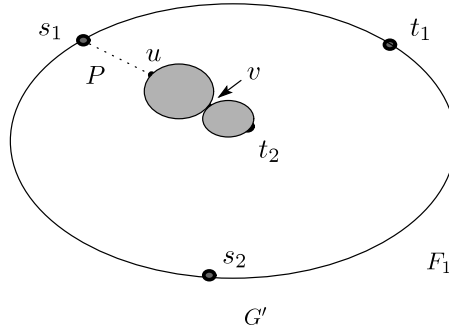


Fig. 3. Construction of  $G'$ .

Let  $s_1, s_2, t_1 \in \partial F_1$  and  $t_2 \in \partial F_2$  be terminals. Let  $P$  be the shortest path connecting  $s_1$  and  $t_2$ . The basic idea of the algorithm is to, for all pairs of vertices  $u, v$  on  $P$ , transform the original problem to an instance of the problem that can be solved using the algorithm by Colin de Verdière and Schrijver. The transformation is described in Lemma 5. In Lemma 4 we prove that the solution remains optimal.

**Lemma 4.** Suppose that a pair of paths  $(P_1, P_2)$  is the unique optimal solution of the Min-Sum 2 Disjoint Paths Problem. Let  $u$  be the vertex in  $V(P_1) \cap V(P)$  closest to  $t_2$  in the ordering along  $P$ , and let  $v$  be the vertex in  $V(P_2) \cap V(P^{[u, t_2]})$  closest to  $u$  in the ordering along  $P$  (Fig. 2). Then,  $P^{[v, t_2]}$  is a subpath of  $P_2$ .

**Proof.** Suppose that  $P^{[v, t_2]}$  is not a subpath of  $P_2$ , and define  $P'_2 = P_2^{[s_2, v]} \cup P^{[v, t_2]}$ . By definition of  $u$  and  $v$ ,  $P_1$  and  $P'_2$  are disjoint. Since  $P$  is the shortest path, every subpath  $P^{[a, b]}$  is the shortest path between  $a$  and  $b$ , thus,  $l(P^{[v, t_2]}) < l(P_2^{[v, t_2]})$ , which implies that  $l(P'_2) < l(P_2)$ . Here we use the fact that the shortest path is unique. Then,  $(P_1, P'_2)$  is a shorter solution, which contradicts the optimality of  $(P_1, P_2)$ .  $\square$

**Lemma 5.** For distinct vertices  $u, v$  on  $P$  such that  $u$  is closer to  $s_1$  than  $v$ , in  $O(n \log n)$  time, we can either find two simple disjoint paths  $P_1$  and  $P_2$  minimizing the total length  $l(P_1) + l(P_2)$  such that

1.  $P_i$  connects  $s_i$  and  $t_i$  for  $i = 1, 2$ ,
2.  $u \in V(P_1)$  and  $V(P_1) \cap V(P) \subseteq V(P^{[s_1, u]})$ , and
3.  $P_2 \cap P^{[u, t_2]} = P^{[v, t_2]}$ ,

or conclude that such  $P_1$  and  $P_2$  do not exist.

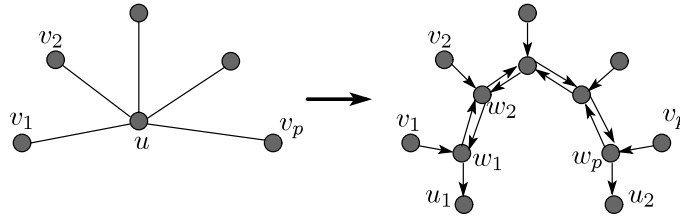
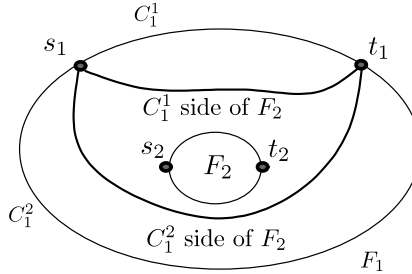
**Proof.** Delete all vertices in  $V(P^{[u, t_2]}) \setminus \{u, v, t_2\}$ . This yields a graph  $G'$  (see Fig. 3). Note that  $u$  and  $v$  are on the boundary of the same face  $F'$  in  $G'$ , because all internal vertices of  $P^{[u, v]}$  have been removed.

We find three paths  $Q_1, Q_2$ , and  $Q_3$  in  $G'$  minimizing the total length such that

- $Q_1$  connects  $s_1$  and  $u$ ,  $Q_2$  connects  $t_1$  and  $u$ ,  $Q_3$  connects  $s_2$  and  $v$ ,
- $V(Q_2) \cap V(Q_3) = V(Q_3) \cap V(Q_1) = \emptyset$ , and  $V(Q_1) \cap V(Q_2) = \{u\}$ .

Then,  $P_1 = Q_1 \cup Q_2$  and  $P_2 = Q_3 \cup P^{[v, t_2]}$  are the desired disjoint paths in  $G$ .

In order to apply Theorem 1, we divide  $u$  into two distinct vertices and construct a digraph as follows. Let  $v_1, v_2, \dots, v_p$  be the vertices in  $G'$  adjacent to  $u$  such that  $v_1, v_p \in \partial F'$  and  $uv_1, uv_2, \dots, uv_p$  are incident to  $u$  in this order. Let  $D_1 = (V_1, E_1)$  be the digraph obtained from  $G - u$  by replacing each edge with two parallel arcs of opposite direction. Define a digraph  $D_2 = (V_2, E_2)$  (see Fig. 4) by

Fig. 4. Construction of  $D_2$ .Fig. 5.  $C_1^1$  side and  $C_1^2$  side.

$$V_2 = V_1 \cup \{w_1, w_2, \dots, w_p, u_1, u_2\},$$

$$E_2 = E_1 \cup \bigcup_{i=1}^p \{(v_i, w_i)\} \cup \bigcup_{i=1}^{p-1} \{(w_i, w_{i+1}), (w_{i+1}, w_i)\} \cup \{(w_1, u_1), (w_p, u_2)\}.$$

Define a new length function  $l' : E_2 \rightarrow \mathbb{R}_+$  as

$$l'(e) = \begin{cases} l(xy) & \text{if } e = (x, y) \text{ or } (y, x) \text{ for } xy \in E, \\ l(v_i u) & \text{if } e = (v_i, w_i), \\ 0 & \text{otherwise.} \end{cases}$$

By finding three disjoint paths  $Q'_1, Q'_2, Q'_3$  with minimum total length such that  $Q'_1$  is from  $s_1$  to  $u_1$  (or  $u_2$ , respectively),  $Q'_2$  is from  $t_1$  to  $u_2$  (or  $u_1$ , respectively), and  $Q'_3$  is from  $s_2$  to  $v$ , we can obtain the desired paths  $Q_1, Q_2$ , and  $Q_3$ . This can be done in  $O(n \log n)$  time by [Theorem 1](#).  $\square$

**Proof of Proposition 3.** By [Lemma 4](#), we can find the optimal solution of the Min-Sum 2 Disjoint Paths Problem by executing the procedure described in [Lemma 5](#) for each pair of vertices  $u$  and  $v$  on the shortest path between  $s_1$  and  $t_2$ . This concludes the proof of [Proposition 3](#).  $\square$

**Proposition 6.** Let  $G = (V, E)$  be an undirected planar graph, and  $F_1$  and  $F_2$  be two of its faces. If  $s_1, t_1 \in \partial F_1$  and  $s_2, t_2 \in \partial F_2$  are terminals, then the Min-Sum 2 Disjoint Paths Problem in  $G$  is solvable in  $O(n^3 \log n)$  time.

**Proof.** Let  $C_i^1$  and  $C_i^2$  be components of  $\partial F_i - \{s_i, t_i\}$  for  $i = 1, 2$ . We say that a path  $P$  connecting  $s_1$  and  $t_1$  (or  $s_2$  and  $t_2$ ) is on the  $C_i^j$  side of  $F_2$  (or on the  $C_i^j$  side of  $F_1$ , respectively) if  $P \cup C_i^j$  (or  $P \cup C_i^j$ , respectively) does not separate  $F_1$  and  $F_2$  for  $j = 1, 2$  ([Fig. 5](#)).

The following lemma directly implies [Proposition 6](#).  $\square$

**Lemma 7.** For  $j_1, j_2 \in \{1, 2\}$ , there is an  $O(n^3 \log n)$  time algorithm to find two paths  $P_1$  and  $P_2$  such that

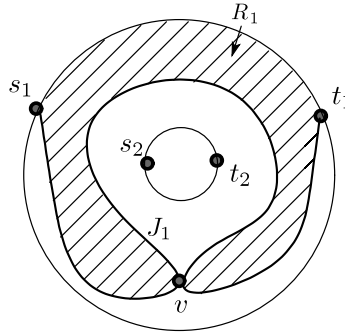
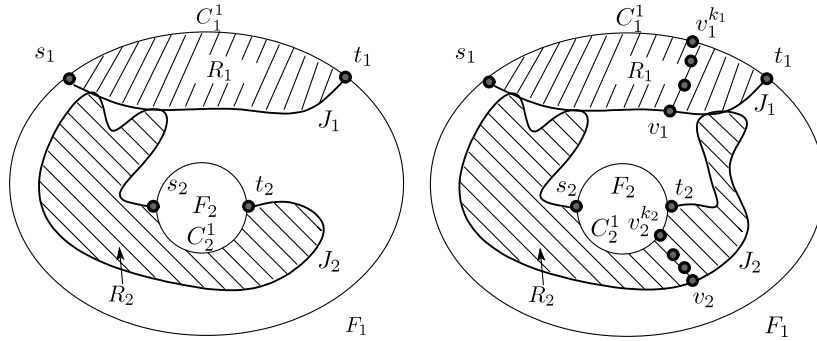
- (1)  $P_i$  connects  $s_i$  and  $t_i$ ,
- (2)  $P_1$  is on the  $C_1^{j_1}$  side of  $F_2$ ,  $P_2$  is on the  $C_2^{j_2}$  side of  $F_1$ , and
- (3) if the optimal solution of the original Min-Sum 2 Disjoint Paths Problem satisfies (1) and (2),  $(P_1, P_2)$  is the optimal solution.

**Proof.** By symmetry, it suffices to consider the case  $j_1 = j_2 = 1$ .

We take the shortest path connecting  $s_i$  and  $t_i$  that is on the  $C_i^1$  side for  $i = 1, 2$ . Note that  $J_i$  is not necessarily simple, but  $J_i$  does not cross with itself (see [Fig. 6](#)) as will become clear later. The inside of  $C_i^1 \cup J_i$  is denoted by  $R_i$ . A precise description of the algorithm of this part is as follows.

**Claim 8.** We can find the shortest path  $J_i$  among all paths connecting  $s_i$  and  $t_i$  that are on the  $C_i^1$  side in  $O(n \log n)$  time.

**Proof of Claim 8.** We only deal with the case  $i = 1$ . The case for  $i = 2$  is analog. We find the shortest paths  $J_s$  from  $s_1$  to  $\partial F_2$  and  $J_t$  from  $t_1$  to  $\partial F_2$ . Then,  $J_s$  and  $J_t$  do not cross. Let  $v_s$  and  $v_t$  be end vertices of  $J_s$  and  $J_t$  in  $\partial F_2$ , respectively. Let  $J \subseteq \partial F_2$  be the path connecting  $v_s$  and  $v_t$  such that  $J_s \cup J \cup J_t$  is on the  $C_1^1$  side of  $F_2$ .

Fig. 6. Case when  $J_1$  is not simple.Fig. 7. Relation between  $F_1'$  and  $F_2'$ .

Then, the desired  $J_1$  is contained in the inside of  $C_1^1 \cup J_s \cup J \cup J_t$ . By finding the shortest path from  $s_1$  to  $t_1$  in the inside of  $C_1^1 \cup J_s \cup J \cup J_t$ , we can find the desired  $J_1$ . This can be done in  $O(n \log n)$  time by using Dijkstra's algorithm [24] in the graph obtained by removing all vertices on the outside of  $C_1^1 \cup J_s \cup J \cup J_t$  and duplicating all the vertices on  $J_s \cap J_t$ .  $\square$

Then, one can see that  $R_1 \cap R_2 \subseteq J_1 \cap J_2$ . In other words,  $J_1$  and  $J_2$  do not cross.

**Claim 9.** Suppose that  $J_1, J_2, R_1, R_2$  are defined as above. Consider the problem of finding the pair of paths minimizing the total length among the pairs of disjoint paths  $(P_1, P_2)$  satisfying (1) and (2) of Lemma 7. If this problem has a feasible solution, then the optimal solution  $(P_1, P_2)$  satisfies that  $P_i \subseteq R_i$  for  $i = 1, 2$ .

**Proof of Claim 9.** Suppose that  $(P_1, P_2)$  is the optimal solution of the problem, and the interior of a path  $P_1^{[u,v]}$  is in  $\Sigma - R_1$  for some  $u, v \in V(P_1) \cap V(J_1)$ .

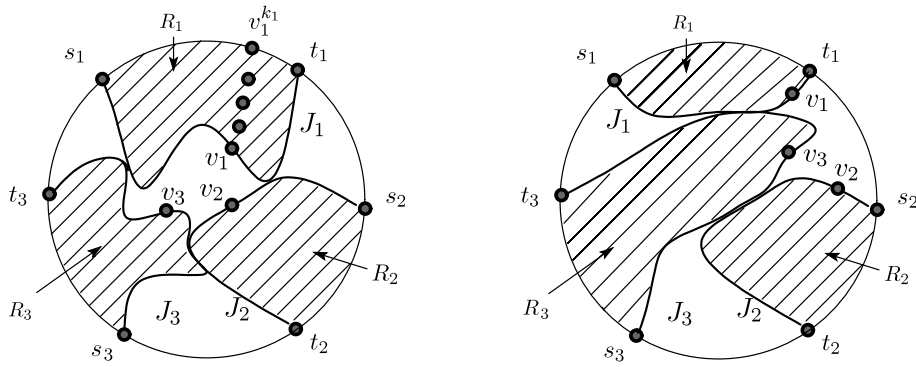
Then, replace  $P_1$  by a new path  $P_1'$  defined by  $P_1' = (P_1 \setminus P_1^{[u,v]}) \cup J_1^{[u,v]}$ . Since  $J_1$  is the shortest path, we have  $l(J_1^{[u,v]}) < l(P_1^{[u,v]})$ , and hence  $l(P_1') < l(P_1)$ . Since  $P_1$  and  $P_2$  are mutually disjoint,  $P_2$  does not intersect with the inside of  $C_1^1 \cup P_1$ . Thus,  $P_1'$  and  $P_2$  are mutually disjoint, because  $P_1'$  is on the inside of  $C_1^1 \cup P_1$ . This contradicts the optimality of  $(P_1, P_2)$ . Hence, we have  $P_1 \subseteq R_1$ , and  $P_2 \subseteq R_2$  is shown in the same way.  $\square$

By this claim, it suffices to find the disjoint paths in  $G[R_1 \cup R_2]$ . We distinguish the following two cases. In the first case, we consider simple paths  $J_i$ . In the second case, we modify  $J_i$  such that we get a simple path, for which the procedure of the first case can be applied.

**Case 1:** First, we consider the case when both  $J_1$  and  $J_2$  are simple. Define  $G' = G[R_1 \cup R_2]$ , and let  $F_1'$  and  $F_2'$  be its faces containing  $F_1$  and  $F_2$ , respectively. Then we can consider the following two cases:  $F_1' = F_2'$  or  $F_1' \neq F_2'$  (see Fig. 7). Since the case  $F_1' = F_2'$  is solvable by a min-cost flow algorithm in  $G'$ , we may assume that  $F_1' \neq F_2'$ .

Then, there exist vertices  $v_1 \in \partial F_2' \cap J_1$  and  $v_2 \in \partial F_1' \cap J_2$ . We take a set of vertices  $v_1^0, v_1^1, \dots, v_1^{k_1}$  in  $R_1$  such that  $P_1$  passes through at least one vertex in this set, and we deal with some cases separately depending on the vertices on  $P_1$ . The sequence  $v_1^0, v_1^1, \dots, v_1^{k_1}$  is constructed as follows:  $v_1^0 = v_1$ ,  $v_1^{k_1} \in C_1^1$  (if  $C_1^1$  consists of one edge, we add a vertex in the middle of the edge, and the added vertex is  $v_1^{k_1}$ ) and  $v_1^l$  is on the boundary of the face of  $G' - ((\delta(v_1^0) \cup \dots \cup \delta(v_1^{l-1})) - R_2)$  containing  $F_2'$ . Then,  $P_1$  passes through at least one of  $v_1^0, \dots, v_1^{k_1}$ . Note that we can use  $G' - \{v_1^0, \dots, v_1^{l-1}\}$  instead of  $G' - ((\delta(v_1^0) \cup \dots \cup \delta(v_1^{l-1})) - R_2)$  if  $v_1^0, v_1^1, \dots, v_1^{l-1} \notin J_2$ . We take  $v_2^0, v_2^1, \dots, v_2^{k_2}$  in the same way. We also note that such sequences exist, because  $J_1$  and  $J_2$  are simple.



Fig. 8. Definitions of  $J_i$  and  $R_i$ .

**Claim 10.** Given  $0 \leq l_1 \leq k_1$  and  $0 \leq l_2 \leq k_2$ , we can find in  $O(n \log n)$  time disjoint paths  $P_1$  and  $P_2$  of minimum total length such that  $P_i$  passes through  $v_i^{l_i}$  but not through  $v_i^0, \dots, v_i^{l_i-1}$  for each  $i$ .

**Proof of Claim 10.** We find four paths  $P_{s_1}, P_{t_1}, P_{s_2},$  and  $P_{t_2}$  in  $G_{l_1, l_2} = G' - ((\delta(v_1^0) \cup \dots \cup \delta(v_1^{l_1-1})) - R_2) - ((\delta(v_2^0) \cup \dots \cup \delta(v_2^{l_2-1})) - R_1)$  minimizing the total length such that  $P_{s_i}$  connects  $s_i$  and  $v_i^{l_i}$ ,  $P_{t_i}$  connects  $t_i$  and  $v_i^{l_i}$ , and they are mutually vertex-disjoint except for  $V(P_{s_i}) \cap V(P_{t_i}) = \{v_i^{l_i}\}$ .

In  $G_{l_1, l_2}$ , the vertices  $v_1^{l_1}, s_2, t_2$  are on the boundary of a common face, and the vertices  $v_2^{l_2}, s_1, t_1$  are on the boundary of another face. Thus, using the same argument as in the proof of Lemma 5, we can find in  $O(n \log n)$  time four such paths by Theorem 1 with  $k = 4$ .

Then,  $P_1 = P_{s_1} \cup P_{t_1}$  and  $P_2 = P_{s_2} \cup P_{t_2}$  are the desired paths.  $\square$

We can easily see that if a path  $P_i$  between  $s_i$  and  $t_i$  is contained in  $R_i$ , then there exists an integer  $0 \leq l \leq k_i$  such that  $P_i$  passes through  $v_i^l$  but not through  $v_i^0, \dots, v_i^{l-1}$ , because  $P_i$  passes through at least one of  $v_i^0, \dots, v_i^{k_i}$ . Therefore, in order to solve the problem in Lemma 7, it suffices to execute the procedure described in Claim 10 for every pair  $(l_1, l_2)$ . Hence, it can be done in  $O(n^3 \log n)$  time.

**Case 2:** Next we consider the case when  $J_i$  is not simple. We assume that  $J_1$  passes through a vertex  $v$  twice, and let  $J'$  be the simple path through  $s_1, v, t_1$  consisting of two subpaths of  $J_1$ . Then,  $J'$  is shorter than  $J_1$  and on the  $C_2^1$  side of  $F_2$  (see Fig. 6). Note that  $J_1$  may touch itself along a contiguous subpath, but in this case we can immediately conclude that the optimal solution of the original Min-Sum 2 Disjoint Paths Problem does not satisfy (2) in Lemma 7, because there exists a path on the  $C_2^2$  side of  $F_2$  which is shorter than  $J_1$  and does not intersect with  $R_2$ .

By Claim 9, when we find the shortest disjoint paths  $(P_1, P_2)$  satisfying (1) and (2) in Lemma 7, we may assume that  $P_2$  and  $J'$  intersect only at  $v$ , or do not intersect. For a pair of paths  $(P_1, P_2)$  satisfying (1) and (2), if  $P_2 \cap J' = \emptyset$ ,  $(P_1, P_2)$  is not the optimal solution of the original Min-Sum 2 Disjoint Paths Problem, because  $(J', P_2)$  is shorter than  $(P_1, P_2)$ .

Hence, we only consider the case when  $P_2$  passes through  $v$ . In this case, we find the shortest path  $J'_1$  from  $s_1$  to  $t_1$  in  $G[R_1] - v$ , and replace  $J_1$  with  $J'_1$ . Then we can execute the same procedure as for Case 1.

This completes the proof of Lemma 7.  $\square$

### 3.2. Min-Sum 3 Disjoint Paths Problem

**Theorem 11.** Let  $G = (V, E)$  be an undirected planar graph and let  $F$  be one of its faces. If all six terminals are on  $\partial F$ , then the Min-Sum 3 Disjoint Paths Problem in  $G$  is solvable in  $O(n^4 \log n)$  time.

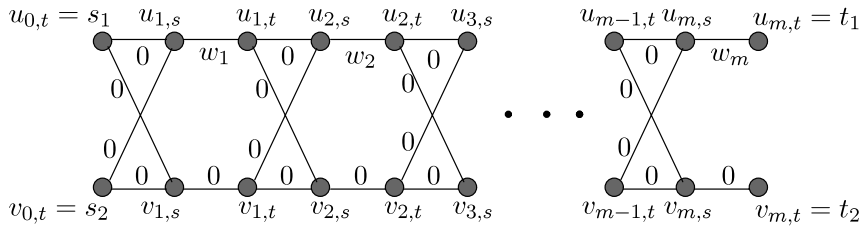
**Proof.** We assume that  $F$  is the outer unbounded face. By exchanging the labels of the vertices, it suffices to consider the case when  $s_1, t_1, s_2, t_2, s_3,$  and  $t_3$  are clockwise in this order along  $\partial F$ . Note that, in the other cases, we can solve the problem by a minimum cost flow algorithm, or we can conclude immediately that there is no feasible solution. Note that we can find the minimum cost flow by simply finding shortest paths three times in the residual graph [25], and hence the running time of this step is  $O(n \log n)$ .

Let  $\partial F_i$  be the component of  $\partial F - \{s_i, t_i\}$  that does not contain any terminals for  $i = 1, 2, 3$ . For each  $i$ , let  $J_i$  be the shortest path connecting  $s_i$  and  $t_i$  in  $G$ , and the inside of  $J_i \cup \partial F_i$  is denoted by  $R_i$ . Then, one can see that  $R_i \cap R_j \subseteq J_i \cap J_j$  for distinct  $i$  and  $j$ . In other words,  $J_i$  and  $J_j$  do not cross (see Fig. 8).

In the same way as Claim 9, we have the following claim.

**Claim 12.** If there exists a feasible solution of the Min-Sum 3 Disjoint Paths Problem, the optimal solution  $(P_1, P_2, P_3)$  satisfies that  $P_i \subseteq R_i$  for  $i = 1, 2, 3$ .



Fig. 9. Graph  $G$  with tree-width 3.

By Claim 12, it suffices to deal with the graph  $G[R_1 \cup R_2 \cup R_3]$ . For convenience, in order to increase the connectivity of the graph, we add to  $G[R_1 \cup R_2 \cup R_3]$  three edges  $t_1s_2$ ,  $t_2s_3$ ,  $t_3s_1$  such that all terminals are on the boundary of the outer unbounded face. Let  $G'$  denote the obtained planar graph.

Then, there exists a bounded face (or a single vertex)  $Q$  of  $G'$  intersecting with  $J_1, J_2$ , and  $J_3$ . Take  $v_i \in V(J_i) \cap Q$  for  $i = 1, 2, 3$  (see Fig. 8).

In a similar way as Case 1 in the proof of Lemma 7, we find a sequence of vertices  $v_1^0, v_1^1, \dots, v_1^{k_1}$  in  $R_1$  such that  $v_1^0 = v_1$ ,  $v_1^{k_1} \in \partial F_1$  and  $v_1^l$  is on the boundary of the face of  $G' - E_1^l$  containing  $Q$ , where  $E_1^l = (\delta(v_1^0) \cup \dots \cup \delta(v_1^{l-1})) - (R_2 \cup R_3)$ . We also find  $v_i^0, v_i^1, \dots, v_i^{k_i}$  and define  $E_i^l$  for  $i = 2, 3$  in the same way.

Next we show the following claim.

**Claim 13.** For integers  $l_1, l_2, l_3$  with  $0 \leq l_i \leq k_i$ , we can find in  $O(n \log n)$  time disjoint paths  $P_1, P_2, P_3$  of minimum total length such that  $P_i$  passes through  $v_i^{l_i}$  but not through  $v_i^0, \dots, v_i^{l_i-1}$  for each  $i$ .

**Proof of Claim 13.** We find six paths  $P_{s_1}, P_{t_1}, P_{s_2}, P_{t_2}, P_{s_3}$ , and  $P_{t_3}$  in  $G' - E_1^{l_1} - E_2^{l_2} - E_3^{l_3}$  minimizing the total length such that  $P_{s_i}$  connects  $s_i$  and  $v_i^{l_i}$ ,  $P_{t_i}$  connects  $t_i$  and  $v_i^{l_i}$ , and they are mutually vertex-disjoint except for  $V(P_{s_i}) \cap V(P_{t_i}) = \{v_i^{l_i}\}$ . Since  $v_1^{l_1}, v_2^{l_2}$ , and  $v_3^{l_3}$  are on the boundary of a same face of  $G' - E_1^{l_1} - E_2^{l_2} - E_3^{l_3}$ , using the same argument as the proof of Lemma 5, we can find in  $O(n \log n)$  time six such paths by Theorem 1 with  $k = 6$ .

Then,  $P_1, P_2, P_3$  defined by  $P_i = P_{s_i} \cup P_{t_i}$  are the desired paths.  $\square$

If a path  $P_i$  between  $s_i$  and  $t_i$  is contained in  $R_i$ , then there exists an integer  $0 \leq l \leq k_i$  such that  $P_i$  passes through  $v_i^l$  but not through  $v_i^0, \dots, v_i^{l-1}$ . Thus, in order to solve the original problem, it suffices to execute the procedure described in Claim 13 for every triple  $(l_1, l_2, l_3)$ , which can be done in  $O(n^4 \log n)$  time.  $\square$

#### 4. Min-Max objective function

We reprove NP-hardness of the Min-Max 2 Disjoint Paths Problem for planar undirected graphs with tree-width 3, using a reduction from the PARTITION problem. We later learned that the reduction was used independently in almost the same manner in [17] already, without an explicit link to the tree-width and the Min-Max variant. For graphs with tree-width 2 (including series-parallel graphs and outer-planar graphs), we provide a polynomial-time algorithm for the Min-Max 2 Disjoint Paths Problem in undirected graphs.

##### 4.1. Hardness

In the PARTITION problem, we are given  $m$  items with weights  $w_1, w_2, \dots, w_m \in \mathbb{Z}_+$ , which are to be split into two subsets of the same weight. The PARTITION problem is also one of Karp's NP-hard problems [5]. Note that the problem is weakly NP-hard, that is, it is NP-hard when the input size of the problem is defined as  $O(m + \log(\max_i w_i + 1))$ . We also note that, using dynamic programming, the problem is solvable in time polynomial in  $m$  and  $\max_i w_i + 1$ . Observe that the hard graph for the Min-Max objective function used in our proof is trivial for the Min-Sum objective function.

**Theorem 14** ([17]). The Min-Max 2 Disjoint Paths Problem is (weakly) NP-hard for planar graphs with tree-width 3.

**Proof.** The polynomial-time reduction from the NP-hard problem PARTITION  $(w_0, w_1, \dots, w_{m-1})$  to a graph is shown in Fig. 9.

For each weight  $w_i$  we add four vertices (two on each side, source and target) and six edges; the graph  $G = (V, E)$  is defined by

$$V := \{u_{0,t}, v_{0,t}\} \cup \bigcup_{i=1}^m V_i, \quad E := \bigcup_{i=1}^m E_i,$$

where

$$V_i = \{u_{i,s}, u_{i,t}, v_{i,s}, v_{i,t}\}, \quad E_i = \{u_{i,s}u_{i,t}, u_{i-1,t}u_{i,s}, u_{i-1,t}v_{i,s}, v_{i,s}v_{i,t}, v_{i-1,t}v_{i,s}, v_{i-1,t}u_{i,s}\}.$$

We define  $s_1 = u_{0,t}$ ,  $s_2 = v_{0,t}$ ,  $t_1 = u_{m,t}$ , and  $t_2 = v_{m,t}$ . All edges have weight 0, except for the edges  $(u_{i,s}, u_{i,t})$ , which have weight  $w_i$ . This graph has tree-width 3 as we can put two consecutive vertices from one side and their counterparts on the other side into a subset, which yields a path as tree-decomposition. Note that the graph can be embedded in the plane without edge intersections.

Two optimal paths with respect to the Min-Max objective function solve the partition problem as follows: if the edge  $u_{i,s}u_{i,t}$  is on path  $P_1$ , include  $w_i$  in set  $S_1$ , otherwise include  $w_i$  in set  $S_2$ . In each ‘step’  $i$ , only one path may use the 0-edge and the other path is required to take the  $w_i$ -edge. The Min-Max objective function is optimized if and only if the weights are partitioned evenly and both paths have the same length.  $\square$

**Corollary 15.** *The Min-Max 2 Disjoint Paths Problem is (weakly) NP-hard for planar graphs.*

We remark here that the above reduction also works for the problem of finding two internally disjoint  $s$ – $t$  paths minimizing the length of the longer path.

#### 4.2. Polynomial-time algorithm for tree-width-2 undirected graphs

We first give a polynomial-time algorithm for outer-planar graphs. Recall that a graph is *outer-planar* if it can be drawn such that every vertex is incident to the outer face and no edges cross. If the graph is not 2-connected, then we can easily reduce the Min-Max 2 Disjoint Paths Problem in the graph to that in a 2-connected graph. So, we only consider 2-connected outer-planar graphs. Consider the cyclic order of terminals in clockwise direction of an outer-planar graph. For  $k \geq 2$  terminal pairs, if there is a feasible solution – that is, there are  $k$  disjoint paths – it may never happen that *only one* terminal of a pair  $(s_i, t_i)$  lies *between* the terminals of another pair  $(s_j, t_j)$ , since their paths  $P_i$  and  $P_j$  would intersect. Without loss of generality, we may assume that the cyclic order of the two terminal pairs is  $(s_1, t_1, t_2, s_2)$ . Note that a minimum cost flow algorithm cannot optimize the Min-Max objective function. We name the sets of nodes between the terminals by *North*, *South*, *East*, and *West* as follows: let  $\mathcal{W}$  denote the nodes between  $s_1$  and  $s_2$ , let  $\mathcal{N}$  denote the nodes between  $s_1$  and  $t_1$ , let  $\mathcal{E}$  denote the nodes between  $t_1$  and  $t_2$ , and let  $\mathcal{S}$  denote the nodes between  $s_2$  and  $t_2$ . Since the graph is 2-connected, these sets are well-defined. Note that since the graph is outer-planar, these sets and the set of terminals comprise all the vertices of the graph. Nodes and edges from and to  $\mathcal{N}$  may not be used by  $P_2$  and, analogously, nodes in  $\mathcal{S}$  are prohibited for  $P_1$ . Of course, every edge may be used by at most one path. Observe that if an edge between a node from  $\mathcal{W}$  and a node from  $\mathcal{E}$  is part of the optimal solution, then all nodes and edges ‘below’ this edge may only be part of  $P_2$  and all edges ‘above’ may only be part of  $P_1$ . We split the problem of finding disjoint paths into a polynomial number of two independent shortest path problems. Every pair of nodes from  $(\{s_2\} \cup \mathcal{W}) \times (\{t_2\} \cup \mathcal{E})$  defines a partition, which defines two vertex-induced subgraphs. The algorithm solves two independent shortest path problems, one for each subgraph.

The running time can be improved by a linear factor if, instead of node pairs, we consider edges between  $\mathcal{W}$  and  $\mathcal{E}$ .

**Algorithm 1.** Input: an undirected 2-connected outer-planar graph  $G = (V, E)$ , terminals  $s_1, s_2, t_1, t_2 \in V$ , w.l.o.g. in cyclic order  $(s_1, t_1, t_2, s_2)$ .

Let  $\mathcal{W}$  denote the nodes between  $s_1$  and  $s_2$ , let  $\mathcal{N}$  denote the nodes between  $s_1$  and  $t_1$ , let  $\mathcal{E}$  denote the nodes between  $t_1$  and  $t_2$ , and let  $\mathcal{S}$  denote the nodes between  $s_2$  and  $t_2$ .

1. For each pair  $(w, e) \in (\{s_2\} \cup \mathcal{W}) \times (\{t_2\} \cup \mathcal{E})$  such that  $we \in E$  or  $(w, e) = (s_2, t_2)$ .
  - Partition  $\mathcal{W}$  into  $\mathcal{W}_1$  containing the nodes between  $w$  and  $s_1$  and  $\mathcal{W}_2 = \mathcal{W} \setminus \mathcal{W}_1$ .
  - Partition  $\mathcal{E}$  into  $\mathcal{E}_1$  containing the nodes between  $e$  and  $t_1$  and  $\mathcal{E}_2 = \mathcal{E} \setminus \mathcal{E}_1$ .
  - Find a shortest path  $P_1$  between  $s_1$  and  $t_1$  in  $G_1 = G[\{s_1, t_1\} \cup \mathcal{N} \cup \mathcal{W}_1 \cup \mathcal{E}_1]$ .
  - Find a shortest path  $P_2$  between  $s_2$  and  $t_2$  in  $G_2 = G[\{s_2, t_2\} \cup \mathcal{S} \cup \mathcal{W}_2 \cup \mathcal{E}_2]$ .
  - Update the previous optimum  $(SP_1, SP_2)$  if  $(P_1, P_2)$  is a better solution.
2. Return the optimal solution  $(SP_1, SP_2)$ .

**Theorem 16.** *Algorithm 1 solves the Min-Max 2 Disjoint Paths Problem for 2-connected outer-planar graphs in time  $O(n^2)$ .*

**Proof (Time Complexity).** The number of edges between  $\mathcal{W}$  and  $\mathcal{E}$  is at most  $O(n)$ . For each pair of endpoints the algorithm solves two independent shortest path problems in time  $O(n)$  [26].

**Correctness:** Each edge may be used by at most one path. If an edge between a node from  $\mathcal{W}$  and a node from  $\mathcal{E}$  is part of the optimal solution, then all nodes and edges ‘below’ this edge and this edge itself may only be part of  $P_2$  and all edges ‘above’ may only be part of  $P_1$ . Therefore, by computing the solution for all possible partitions, the algorithm finds the optimal solution.  $\square$

In what follows, we reduce the Min-Max 2 Disjoint Paths Problem in graphs with tree-width 2 to the problem in outer-planar graphs.

**Definition 2.** A graph has *contracted tree-width* 2 if it has tree-width 2 and the corresponding tree-decomposition  $(T, \mathcal{V})$  satisfies that no pair of vertices is contained in more than two vertex sets of  $\mathcal{V}$ , that is  $|V_{t_1} \cap V_{t_2} \cap V_{t_3}| \leq 1$  for any  $t_1, t_2, t_3 \in V(T)$ .

**Lemma 17.** *Graphs with contracted tree-width 2 are outer-planar.*

**Proof** (*Proof is by induction*). A graph consisting of three vertices is obviously outer-planar. Let  $(T, \mathcal{V})$  be the tree-decomposition of  $G = (V, E)$  with contracted tree-width 2. We may assume that  $V_i \neq V_j$  for distinct  $i, j \in V(T)$ . Take a leaf  $l \in V(T)$  of  $T$  and let  $V' = \bigcup_{t \neq l} V_t$ . Then, by induction hypothesis,  $G[V']$  is outer-planar. If  $|V_l \cap V'| = 1$  then  $G$  is obviously outer-planar, and so we may assume that  $V_l \cap V' = \{u, v\}$ .

If there exists an edge connecting  $u$  and  $v$  on the boundary of  $G[V']$ , then  $G$  is outer-planar. Similarly, if  $G[V'] \cup \{uv\}$  is an outer-planar graph such that  $uv$  is on the boundary, then  $G$  is outer-planar. Otherwise, there exists a cycle  $C$  such that  $u$  and  $v$  are on  $C$  and  $uv$  is not in  $C$ . Then, by the definition of the tree-decomposition, either  $\{u, v\}$  is part of two sets of  $\mathcal{V} \setminus V_l$ , or no set of  $\mathcal{V} \setminus V_l$  contains  $\{u, v\}$ . The former case contradicts the definition of contracted tree-width, and the latter case contradicts that  $l$  is a leaf of  $T$ .

This shows that  $G$  remains outer-planar.  $\square$

We reduce the problem for a general graph with tree-width at most 2 to equivalent problems in a graph with contracted tree-width 2 (which is also outer-planar), which we then solve using [Algorithm 1](#).

As the tree-width is at most 2, every set  $V_t \in \mathcal{V}$  has cardinality at most three. The graph induced by  $V_t$  and the edges of the original graph may consist of one or two edges or it may form a triangle. We may assume that the overlap with another set  $V_{t'}$  consists of at most two vertices,  $|V_t \cap V_{t'}| \leq 2$  and all these small graphs are linked forming a tree. In the tree-decomposition  $(T, \mathcal{V})$ , we update the length of all edges  $uv$  that lie in more than two sets in  $\mathcal{V}$  by deleting a subgraph or by renaming the terminals.

**Algorithm 2.** Input: a graph  $G = (V, E)$  with tree-width at most 2, terminals  $s_1, s_2, t_1, t_2$ .

1. Compute a tree-decomposition  $(T, \mathcal{V})$  of  $G$  (see [27]). We may assume that  $V_i \neq V_j$  for distinct  $i, j \in V(T)$ .
2. While there exist pairs  $\{u, v\}$  with  $\exists V_{i_1}, V_{i_2}, V_{i_3}, i_1 \neq i_2 \neq i_3 \neq i_1, \{u, v\} \subseteq V_{i_1} \cap V_{i_2} \cap V_{i_3}$  compute  $G - \{u, v\}$  in which there are at least three distinct components  $G_1, G_2, \dots, G_p$ .
  - If some component  $G_i$  contains no terminal, remove  $G_i$ , add an edge  $uv$  (if  $uv$  does not exist), and update  $l(uv) \leftarrow \min\{l(uv), d_{G_i+u+v}(u, v)\}$ .
  - If all  $G_1, G_2, \dots, G_p$  contain terminals, then  $p \in \{3, 4\}$  and two subgraphs, say  $G_1, G_2$ , contain one terminal each. Rename the terminals such that the terminal in  $G_1$  is  $s_1$ . Compute  $d_{G_1+u}(s_1, u)$  and  $d_{G_1+v}(s_1, v)$  and do the same for the terminal in  $G_2$ . Remove  $G_1$  and  $G_2$  and create new instances as follows.
 

If the terminals ‘match’, meaning that the terminal in  $G_2$  is  $t_1$ , recursively create four instances of the problem ( $u = s_1, v = t_1; u = t_1, v = s_1; v = s_1 = t_1$ ; and  $u = s_1 = t_1$ ) and return the minimal solution. Otherwise, if the terminals do not match, recursively create two instances of the problem and return the minimal solution.
3. The resulting graph is outer-planar. Reduce the problem to that in a 2-connected graph, and solve the problem using [Algorithm 1](#).

**Theorem 18.** *Algorithm 2 solves the Min-Max 2 Disjoint Paths Problem for graphs with tree-width at most 2 in time  $O(n^2)$ .*

**Proof** (*Correctness*). Let  $V_{i_j} = \{u, v, w_j\}$  for  $j = 1, 2, 3$ . If  $w_1$  and  $w_2$  are in the same component of  $G - \{u, v\}$ , then  $G$  contains a  $K_4$  (complete graph with 4 vertices) as a minor, which contradicts that the tree-width is at most 2. Similarly,  $w_1, w_2$ , and  $w_3$  are in different components of  $G - \{u, v\}$ , that is, removing  $\{u, v\}$  splits the graph into at least three component graphs.

- If one component, say  $G_i$ , does not contain a single terminal  $s_1, s_2, t_1, t_2$ , update the edge length of  $uv$  by  $\min\{l(uv), d_{G_i+u+v}(u, v)\}$ . This works since only one path may pass through  $G_i$ .
- If  $G_i$  contains exactly one terminal, say  $s_1$ , remove  $G_i$  from  $G$  and solve the disjoint shortest path problem for two separate instances  $u = s_1$  and  $v = s_1$ , add  $d_{G_i+u}(s_1, u)$  and  $d_{G_i+v}(s_1, v)$ , respectively, and return the minimum solution. This works since  $P_2$  cannot enter and leave  $G_i$  without interfering with  $P_1$ .
- If  $G_i$  contains two terminals, then reduce another subgraph  $G_j$  using the rules from above. At most two subgraphs may have this property and the resulting graph will have the desired property for the pair  $\{u, v\}$ .
- The remaining cases are symmetric.

After Step 2, no edge  $uv$  is in more than two sets  $V_i, V_j$  of the tree-decomposition, which is exactly the definition of contracted tree-width 2. The resulting graph is outer-planar by [Lemma 17](#). By computing the solution for all possible pairs, the algorithm finds the optimal solution.

Time complexity: There will be at most  $O(n)$  reductions as there are at most  $O(n)$  edges. The second reduction in Step 2 of [Algorithm 2](#) occurs at most twice, since two terminals in the new instances are  $u$  and  $v$ , which are always in the same component in the algorithm for the new instances. Finding the shortest path in each component  $G_i$  takes time at most  $O(|V(G_i)|)$  [22]. Therefore, all reductions take time at most  $O(n)$  in total. The recursive call occurs at most twice with at most four instances each. This yields at most  $4 \cdot 4 = O(1)$  disjoint shortest path problems in an outer-planar graph, each of which can be solved in time  $O(n^2)$ .  $\square$

#### 4.3. Pseudo-polynomial-time algorithm for bounded tree-width graphs

As shown in [Theorem 14](#), the Min-Max  $k$  Disjoint Paths Problem is NP-hard even if  $k = 2$  and the tree-width of the input graph is at most three, whereas the Min-Sum  $k$  Disjoint Paths Problem can be solved in polynomial time in bounded

tree-width graphs [14]. In this subsection, for fixed  $k$ , we give a pseudo-polynomial-time algorithm for the Min-Max  $k$  Disjoint Paths Problem in bounded tree-width graphs. Note that this technique also works for the weighted versions introduced in [17,19].

**Theorem 19.** *Let  $G = (V, E)$  be a graph whose tree-width is bounded by a fixed constant, and let  $\ell : E \rightarrow \mathbb{Z}_+$  be an integer-valued length function. Then, for fixed  $k$ , the Min-Max  $k$  Disjoint Paths Problem can be solved in time polynomial in  $|V|$  and  $L = \sum_{e \in E} \ell(e)$ .*

**Proof.** We introduce a new problem called the *weighted folio*, whose unweighted version is introduced in [21]. Let  $G = (V, E)$  be a graph, let  $\ell : E \rightarrow \mathbb{Z}_+$  be an integer-valued length function, and let  $X \subseteq V$  be a vertex set. A pair  $(\mathcal{X}, \bar{z})$  of a partition  $\mathcal{X} = \{X_1, X_2, \dots, X_p\}$  of  $X$  and an integer vector  $\bar{z} = (z_1, z_2, \dots, z_p) \in \mathbb{Z}_+^p$  is *realizable* if there are disjoint trees  $T_1, T_2, \dots, T_p$  in  $G$  such that  $X_i \subseteq V(T_i)$  and  $\sum_{e \in E(T_i)} \ell(e) = z_i$  for  $i = 1, \dots, p$ . The weighted folio is the problem to enumerate all realizable pairs  $(\mathcal{X}, \bar{z})$  in  $G$ . One can see that the solution of the Min-Max  $k$  Disjoint Paths Problem is immediately derived from that of the weighted folio in which  $X = \{s_1, \dots, s_k, t_1, \dots, t_k\}$ . That is, for a partition  $\mathcal{X} = \{\{s_1, t_1\}, \dots, \{s_k, t_k\}\}$  of  $X$ , it suffices to find a realizable pair  $(\mathcal{X}, \bar{z})$  minimizing  $\max_i(z_i)$ . Note that the solution of the weighted folio with respect to  $X$  consists of at most  $(|X|L)^{|X|}$  realizable pairs, which is polynomial in  $L$  if  $|X|$  is fixed.

If the tree-width of the input graph is bounded by  $w$ , the weighted folio can be solved for each bag using the standard dynamic programming technique, which takes time polynomial in  $|V|$ ,  $(w + |X|)^{w+|X|}$ , and  $L^{w+|X|}$  (see [22,21]). This completes the proof.  $\square$

## Acknowledgements

The first author is supported by JSPS Research Fellowships for Young Scientists. His work was partially supported by the Global COE Program “The research and training center for new development in mathematics”, MEXT, Japan.

## References

- [1] András Frank, Packing paths, cuts and circuits—a survey, in: *Paths, Flows, and VLSI-Layout*, Springer-Verlag, 1990, pp. 49–100.
- [2] Neil Robertson, Paul D. Seymour, An outline of a disjoint paths algorithm, in: *Paths, Flows, and VLSI-Layout*, Springer-Verlag, 1990, pp. 267–292.
- [3] Richard G. Ogier, Vladislav Rutenburg, Nachum Shacham, Distributed algorithms for computing shortest pairs of disjoint paths, *IEEE Transactions on Information Theory* 39 (2) (1993) 443–455.
- [4] Anand Srinivas, Eytan Modiano, Finding minimum energy disjoint paths in wireless ad-hoc networks, *Wireless Networks* 11 (4) (2005) 401–417.
- [5] Richard M. Karp, On the computational complexity of combinatorial problems, *Networks* 5 (1975) 45–68.
- [6] James F. Lynch, The equivalence of theorem proving and the interconnection problem, *SIGDA Newsletter* 5 (3) (1975) 31–36.
- [7] Alexander Schrijver, Finding  $k$  disjoint paths in a directed planar graph, *SIAM Journal on Computing* 23 (4) (1994) 780–788.
- [8] Steven Fortune, John E. Hopcroft, James Wyllie, The directed subgraph homeomorphism problem, *Theoretical Computer Science* 10 (1980) 111–121.
- [9] Paul D. Seymour, Disjoint paths in graphs, *Discrete Mathematics* 29 (1980) 293–309.
- [10] Yossi Shiloach, A polynomial solution to the undirected two paths problem, *Journal of the ACM* 27 (3) (1980) 445–456.
- [11] Carsten Thomassen, 2-linked graphs, *European Journal of Combinatorics* 1 (1980) 371–378.
- [12] Neil Robertson, Paul D. Seymour, Graph minors. XIII. The disjoint paths problem, *Journal of Combinatorial Theory, Series B* 63 (1) (1995) 65–110.
- [13] Éric Colin de Verdière, Alexander Schrijver, Shortest vertex-disjoint two-face paths in planar graphs, in: *STACS, 2008*, pp. 181–192.
- [14] Petra Scheffler, A practical linear time algorithm for disjoint paths in graphs with bounded tree-width, Technical Report 396, Technische Universität Berlin, 1994.
- [15] Alon Itai, Yehoshua Perl, Yossi Shiloach, The complexity of finding maximum disjoint paths with length constraints, *Networks* 12 (1981) 277–286.
- [16] Chung-Lun Li, S. Thomas McCormick, David Simchi-Levi, The complexity of finding two disjoint paths with min-max objective function, *Discrete Applied Mathematics* 26 (1) (1990) 105–115.
- [17] Bing Yang, Si-Qing Zheng, Enyue Lu, Finding two disjoint paths in a network with normalized  $\alpha^-$ -min-sum objective function, in: *IASTED International Conference on Parallel and Distributed Computing and Systems, 2005*, pp. 342–348.
- [18] Bing Yang, Si-Qing Zheng, Suresh Katukam, Finding two disjoint paths in a network with min-min objective function, in: *IASTED International Conference on Parallel and Distributed Computing and Systems, 2003*, pp. 75–80.
- [19] Bing Yang, Si-Qing Zheng, Enyue Lu, Finding two disjoint paths in a network with normalized  $\alpha^+$ -min-sum objective function, in: *ISAAC, 2005*, pp. 954–963.
- [20] Rudolf Halin,  $S$ -functions for graphs, *Journal of Geometry* 8 (1–2) (1976) 171–186.
- [21] Neil Robertson, Paul D. Seymour, Graph minors. II. Algorithmic aspects of tree-width, *Journal of Algorithms* 7 (1986) 309–322.
- [22] Stefan Arnborg, Andrzej Proskurowski, Linear time algorithms for NP-hard problems restricted to partial  $k$ -trees, *Discrete Applied Mathematics* 23 (1) (1989) 11–24.
- [23] Herbert Edelsbrunner, Ernst Peter Mücke, Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms, *ACM Transactions on Graphics* 9 (1) (1990) 66–104.
- [24] Edsger Wybe Dijkstra, A note on two problems in connection with graphs, *Numerische Mathematik* 1 (1959) 269–271.
- [25] Jack Edmonds, Richard M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, *Journal of the ACM* 19 (2) (1972) 248–264.
- [26] Monika Rauch Henzinger, Philip N. Klein, Satish Rao, Sairam Subramanian, Faster shortest-path algorithms for planar graphs, *Journal of Computer and System Sciences* 55 (1) (1997) 3–23.
- [27] Hans L. Bodlaender, A linear-time algorithm for finding tree-decompositions of small treewidth, *SIAM Journal on Computing* 25 (6) (1996) 1305–1317.